MobCache: A Group-based Caching for Mobile Database Systems

KSHAMA RAICHURA
Shree M. and N. Virani Science College, Gujarat, INDIA
and
NILESH PADHARIYA
Atmiya Institute of Technology & Science, Gujarat, INDIA

This work proposes a group-based caching for Mobile Database System (MDS), denoted as **MobCache** system. In recent era, the proliferation of mobile devices inspiring researchers to bring their capabilities at every field of life. In addition to this, mobile devices like PDAs, smart phones and tablets are capable to store adequate amount of data in its internal memory as per applications' requirements. This enables mobile applications to use internal memory of mobile device as system cache to improve performance of collaborative mobile applications. Today's exponential growth of data and their high access inspired us to develop a group-based caching mechanism for MDSs, hence to reduce the workload on central database server and to cut down a network bandwidth usage. The main contributions of our work are three-fold. First, we propose a group-based caching system for MDSs, designated as MobCache. Second, we propose three different mobile caching schemes in MobCache: (i) Cache-Less MDS (CLM), (ii) Cache-Enabled MDS (CEM) and (iii) Group-based Caching in MDS (GCM). Third, our performance evaluation shows that the MobCache is indeed effective in terms of reducing query response time and the minimizing communication cost, thereby improving query processing in MDS.

Keywords: Sensor networks, ad hoc networks, Routing in sensor networks

1. INTRODUCTION

Today's mobile world brings the power of mobile systems in various areas of society. This opens the new directions for researchers in various areas like mobile education system (i.e., M-Learning), Mobile business system (i.e., M-Commerce), Mobile healthcare system (i.e., M-health); and so on. The recent development in mobile devices provides a large memory on mobile devices itself, hence to bring a data storage on tiny hand-held devices. Furthermore, mobile devices are also capable to deploy database system at smart-scale level on itself. This further inspires technology to bring central server data on mobile clients. Such mobile database systems (MDSs) are widely known as mobile databases. Moreover, the network capabilities like WiFi/Bluetooth enable mobile devices to create an ad-hoc networks on-the-go, which is defined as Mobile Ad-hoc Networks (MANETs).

The encouragement behind the usage of PDA, laptop and cell phone is wireless technology and their supportive applications. In mobile environment, mobile support stations are the medium for mobile clients to retrieve the data. These mobile devices can exchange data with each other or with the centralized entity through the channels like broadcast, multicast, point-to-point or their combinations.

Central Database Server (CDS) facilitates mobile devices to keep live connections with it towards data obtaining. This causes high network traffic and more number of data connections on server, thereby degrading performance of MDS. If system facilitates to cache the data from CDS on tiny devices, then network traffic may be reduced drastically. Furthermore, ad-hoc sharing of the data across the mobile devices provides faster and easy data-access to the mobile clients, while reducing live connections on CDS. Hence, for the performance improvement in data retrieval of mobile clients, caching is an important technique w.r.t. server-data access.

The core challenges in mobile environment are low bandwidth, frequent disconnections, limited power and mobility cause the high network connections with CDS. This raises that how data can be effectively retrieved and managed in mobile environment. The data management in mobile environment merely deals with location-based data management i.e., the data are more relevant according to the locations of mobile devices. Moreover, data replication of a data query on mobile devices and their sharing highly improves the data processing and query response time. In related work section, we have discussed various proposed caching mechanisms such as cooperative caching, semantic caching, group-based caching along with cache-replacement and cache-consistency techniques in distributed environments.

The concept of caching plays an important role in effective working of MDS. Limited energy, bandwidth asymmetricity, frequent disconnections, limited resources are the major constraints in mobile environment. Hence, to provide the continuous services to the mobile-clients, caching plays a key role in mobile data management. It is useful to reduce (i) network traffic on CDS by storing frequently used data on the cache memory of mobile devices, (ii) the communication cost and (iii) the processing time of the queries sent by mobile devices. Recently, there are some parallel works, which have been done in this direction to improve the data availability, the query response time and the query hop-counts in M-P2P networks at reasonable communication cost [Padhariya et al. 2013]. Moreover, an economic model for top-k query processing in M-P2P networks [Padhariya et al. 2011], which helps to improve the performance by minimizing communication traffic.

Lets us consider a medical information (i.e., representation and distribution) system. Here, we consider that the system has a set of medical representatives (MRs), a set of regional heads (RHs) and the core central database server (CDS). The people in society act as mobile peers (MPs) in the system, who query to their nearby MRs for the information about the different deceases, current medical trends, viral infections, various medicares and methodologies, medical preparations etc. Hence, MRs are responsible to bring the information from CDS and distribute the data among mobile peers on-demand. Each MR keeps a small database on its mobile device, which is capable to answer the query sent by a *nearby* mobile user.

In case, that MR does not have query related data, then it forwards the query to its *nearby* group-MRs WiFi / Blutooth / Mobile Internet, although they are geographically located at different regions. Notably, MRs act as small caching-databases. The required data-cache is obtained by MRs from their respective RHs. Each MR is associated with at least one RH, which can also be a mobile entity with high capacity (in terms of processing, storage and communication) mobile device such as tablet or laptop. The high capacity storage of RH is used to keep large number of data-caches, which can be provided to MRs on-demand. Finally, RHs synchronize their data-cache with CDS towards distributing latest information in medical information system on periodic-basis. Observe that how the data can be distributed and cached at different levels (e.g., MR, RH etc.) in system towards effectively answer the mobile user's query in time, while reducing the communication cost and the CDS overhead.

Another application scenario is related to educational information system. Various kinds of information related to subjects, students, faculties, staff members, management, etc. should be effectively manged and propagated across the system. For example, the information related to student like attendance, subject-marks, personal data, activity data, etc. need to be distributed by respective head of departments via their in-charged course-faculty. Hence, the distributed information system helps to disseminate data via caching technique in MDS. Here, faculties, department heads and principal may act as caching-devices while students act as mobile clients, who query the system. The query is answered by *nearby* faculty member, who has the data-cache about his/her classroom. This reduces the load on CDS. Faculties can synchronize their data-caches with CDS, when they are in their comfort zones e.g., department, college-canteen, etc.

The data-caching at different levels improves the overall system performance in MDSs, while reducing the communication overhead across the caching-levels and CDS. Hence, our work focuses on effective data-caching in mobile environment to support mobile databases. The main contributions of our wok are three-fold:

(1) We propose a group-based caching system for MDSs, designated as MobCache.
(2) We propose three different mobile caching schemes in MobCache: (i) Cache-Less MDS (CLM), (ii) Cache-Enabled MDS (CEM) and (iii) Group-based Caching in MDS (GCM).
(3) Our performance evaluation shows that the MobCache is indeed effective in terms of reducing query response time and the minimizing communication cost, thereby improving query processing in MDS.

The reminder of this paper is organized as follows: Section 2 represents related work. Section 3 and Section 4 describe architecture of MobCache and data-caching schemes for MDS respectively. Our performance study is presented in Section 5. Finally, we conclude our work with future direction in Section 6.

2. RELATED WORK

This section discusses the existing work on caching mechanisms and proposals in distributed environment. These proposals are closed to our work.

[Kumar 2006] defines a mobile database system (MDS) as a distributed environment, where a database is hosted on a single mobile host or a group of mobile hosts. In case of group-hosting, the mobile hosts retrieves the data via distributed query and communication with each other using short-range technologies such as WiFi or Bluetooth. In mobile environment, data is commonly shared either using push-mechanism (i.e., data dissemination) or pull-mechanism (i.e., data hosting). The survey of the proposal [Barbará 1999] shows that how the push/pull mechanisms are being performed over limited bandwidth channels, while queried data are location-dependent. Furthermore, the proposals [Chan and Roddick 2003; Tolia et al. 2007; Qureshi et al. 2010] discuss the research issues in MDSs like data retrieval, efficient and optimal database designing, low network bandwidth; and frequent network partitioning (which is caused by mobility and energy constrained mobile peers). Moreover, the proposal [Madria and Bhowdrick 2001] also addresses the data management issues in mobile environment and provides various resolutions like data replication, effective query processing, etc.

Cooperative and semantic caching

The concept of *cooperative* caching enables mobile peers (MPs) to retrieve data from their neighboring MPs via multi-hop channels in mobile networks. The work in [Chow et al. 2004] proposes a COCA model for distributed cooperative caching in M-P2P environment. In COCA, high-activity MPs obtains the replica of a data item from low-activity MPs to provide effective data caching in mobile networks, while system performs load-balancing across MPs. This minimizes the server work load as MPs get benefit of replicas for their respected activity. The work in [Dimokas et al. 2011] proposes two new cooperative caching protocols in wireless sensor networks. The proposed schemes are based on selection metrics for appropriate sensor node selection in order to serve data in short latency with minimum energy.

The notion of *semantic* caching considers the data caching at places of their significance, hence to bring data more closure to the query about that data. In mobile environment, the spatial data needs such semantic caching to effectively utilize the network bandwidth as it is one of the crucial factors in mobile networks. The work in [Sun et al. 2005] proposes a Multi-resolution Semantic Caching (MSC) model by combining multi-resolution spatial data structure with semantic caching techniques in order to effectively process spatial queries. Moreover, the work in [Li et al. 2012] extends the traditional semantic cache management in following three ways: (a) extension of quadtree-based index structures to semantic caches, (b) availability of a query processing strategy and (c) discussion on object-oriented implementation of the semantic cache.

Our work combines both the notions of cooperative and semantic caching to improve the data availability in MDS, while reducing the communication with CDS, thereby shorten query response time in resource-constrained environment.

Group-based caching

Group-based caching is a flavour of cooperative caching with the restrictions based on group-policies. The proposal [Cheng et al. 2007] presents an interest group-based collaborative caching system IntraCache, where group is formed based on the common interests of the peers in web caching system. Although, IntraCache provides efficient search within P2P search space, it restricts the caching and sharing of data only among the group members. The work in [He et al. 2010] provides the effective mechanism to perform the cache replacement based on the cache-groups in microprocessor. The low-accessed cache is replaced within the group by high-required cache, hence to improve in data availability across the group, thereby reducing latency in query processing.

Similarly, a group-based caching approach is proposed in [Amer et al. 2002]. This work creates the groups based on similar files, hence the cache is equivalent to each file-group. This is useful while similar data has been accessed by several peers in network. This is close to our medical information system application scenario, where MRs with similar information form the group and share the same cache across their regions to quickly respond customers' query.

The work in [Chow et al. 2005] proposed group-based cooperative caching scheme for distributed environment by providing hint to the group members via cache signature in order to enhance over all data availability. While extended scheme GroCoca (GROup-based COoperative Caching) [Chow et al. 2007] shows that the caching sign is helpful to determine whether a particular group member can cached data items or not. One more group-based cooperative caching model, called GCC, is proposed in [Ting and Chang 2013]. GCC allows mobile nodes to form a group with neighboring peers in order to exchange a bitmap dictionary periodically.

Cache replacement strategies and cache consistency

Mobile devices have limited primary memory, which causes that the cache-size should be smaller in MDSs as compared to distributed environment. The smaller cache-size indeed requires the effective cache management to improve the system performance in terms of data access and data sharing. Moreover, smaller cache-size causes high cache replacement and their consistency across the mobile hosts.

For effective performance of database systems in distributed environment, maintenance of cache-freshness is necessity, for which the set of data items should be evicted periodically from the cache. The proposals [Kumar et al. 2007] and [Kumar et al. 2010] provide the region-based cache-replacement policies respectively based on weight and priority. The work in [Hu et al. 2005] represents a proactive caching model, which helps the object re-usability as it caches result object along with supported index. While considering the network density, network distance and the probability of access, [Janef et al. 2008] proposes a cache replacement policy, which uses Progressive Incremental Network Expansion (PINE) technique to calculate the network distance.

The work in [Jung et al. 2002] proposes a caching policy and broadcast scheme, which is suitable for urban area by considering the moving distance of mobile host. In the process of hand off, the new server will not get benefit to access the cache. As a solution to this, [Peng and Chen 2005] discovers numerous cache retrieval schemes to improve the cache retrieval efficiency. Moreover, an adaptive per-user per-object cache consistency management (APPCCM) scheme is proposed in [Li and Chen 2011] to support strong data consistency semantics through integrated cache consistency and mobility management in wireless mesh network.

For maintaining cache-consistency in the system, the server broadcasts invalidation report with updated data object so that the mobile clients remove the old data from cache. The work in [Kang and Lim 2001] proposes a set of new cache validation schemes, which are capable of conserving the bandwidth for cache validation as well as for query processing. Moreover, the content in [Zhang et al. 2006] presents a category of cache invalidation strategy and mathematical model in order to develop a high-performance caching technique. Moreover, [Chung 2008] proposes a pradicate-based cache invalidation scheme for continuous partial query in mobile computing environment as the cache state of mobile client is predicate. Additionally, the work in [Sourlas et al. 2009] introduces two caching policies: basic caching and leaf caching for providing guaranteed message delivery.

3. MOBCACHE: A GROUP-BASED CACHING FOR MOBILE DATABASE SYSTEMS

This section discusses our proposed MobCache system for group-based caching in mobile database system (MDS).

Architecture of MobCache

The architecture of MobCache consists of mobile devices (mobile phone, laptop, PDA, tab, etc.) act as mobile peers (MPs) and central database server (CDS) as caching server. In MobCache, MPs have one of the following three roles: **query-issuer (QI), cache-host (CH)** and **cache-manager (CM)**. Each role has independent functions for query processing in mobile networks to retrieve data from CDS. QI issues a query Q through broadcast and retrieves a data as answer

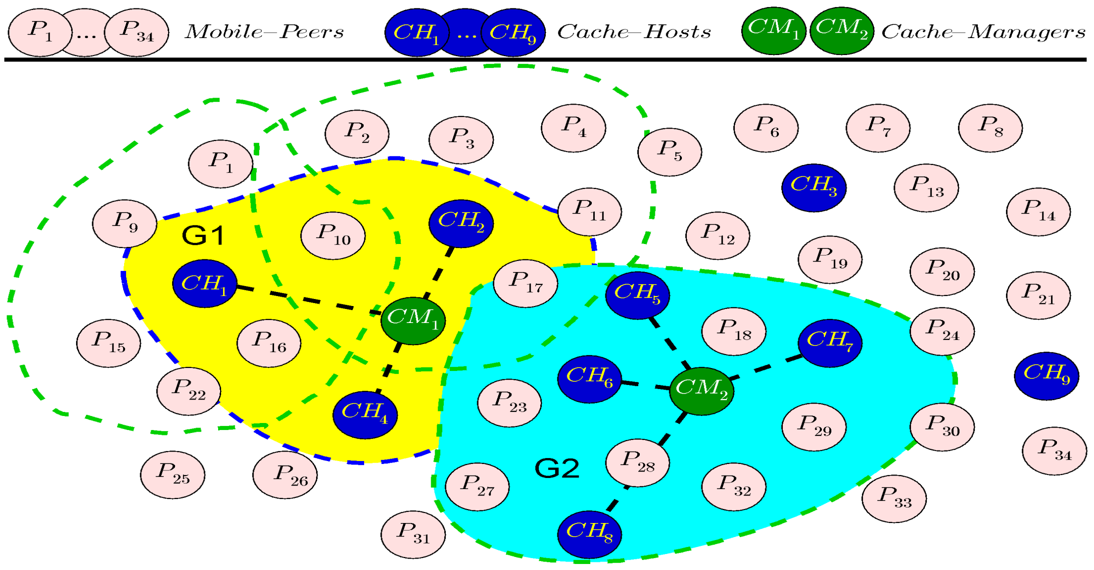

Figure 1: Query Processing in MobCache

of Q from its *nearby* CH. Cache-host performs the following functions: (i) responds the query raised by QI, (ii) stores required data-cache into its device memory in order to respond queries, (ii) maintains data-cache freshness by retrieving data through its relative CM from CDS and (iv) exchanges data-cache with other cache-hosts in its group. Cache-manger is responsible for (i) data-cache synchronization between cache-host and caching server, (ii) providing data-cache to the cache-hosts of its respective group and (iii) buffering hot data-caches (i.e., a frequently required data-cache) from caching-server in order to provide data-cache as and when required by its relative cache-hosts.

MobCache comprises the notion of a group-based caching. A group is a collection of cache-hosts and a cache-manager. They act in a group towards providing queried data at shorter time at reduced communication cost. For example, in our medical information system application scenario, medical representatives (MRs) are the cache-hosts and region-heads (RHs) are the cache-managers, who collaboratively provide the batter service by caching a part of a data at different time with different requirements of mobile users.

Each group member i.e. cache-host associated with only one cache-manager. Note that the groups are isolated and have been created based on application scenario. Hence, cache-hosts CH are the group members, while cache manager CM is a group-leader. CHs are able to communicate with each other via Bluetooth / WiFi / Mobile Internet, while they communicate with cache-manager via WiFi / Mobile Internet. Notably, the high speed data transfer is only required between CHs and their respective CM; or among CHs to transfer cache quickly, while mobile users do not need high-speed internet as they communicate with CHs via short-range technologies such as Bluetooth / WiFi.

We assume that CM has more storage w.r.t. CH in order to store the large pool of cache, but it does not have full data-cache from CDS. CM stores most frequently asked data across all CHs. While CH has smaller cache storage to store relevant data-cache in its own region. Thus, overall objective of group formation is to provide efficient query processing to end users, while minimizing communication cost and reducing processing time on CDS, thereby providing effectiveness of group-based caching in MDS.

Figure 1 depicts the architecture of MobCache. P_1, P_2, ...,P_{34} are MPs, who query in mobile database network by playing a role of query-issuer QI. MPs with medium resource capacity are denoted as cache-hosts (CH_1 ,CH_2, CH_3,...,CH_9), while MPs CM_1 and CM_2 are the cache-mangers in given MDS. Figure 1 also presents two groups G1 and G2, which respectively consist group members (CH_1, CH_2 , CH_4, CM_1) and (CH_5, CH_6, CH_7, CH_8, CM_2). Furthermore, CH_1 serves their nearby MPs (P_1, P_9, P_{10}, P_{15}, P_{16}, P_{22}), while CH_2 serves its nearby (P_2, P_3, P_4, P_{10}, P_{11}, P_{17}) and so on.

In MobCache, we assume *data-cache* is a data available with cache-hosts or cache-manager to answer the query sent by QI. Here, we design a data-cache as a partial database table (location-based) to answer database related query. For example, QI issued a query to find a list of medicines available for 'cold' solution. Then, CH obtains the data-cache in form of data table $< T >(< c_1 >, < c_2 >, ...,< c_n >)$ e.g. medicine (decease-name, vendor, price, shop name, dose, remarks). This database may contain list of medicines including deceases like cold, headache, fever, etc. Thus, if any other QI issues a query related to T e.g., the list of medicines for fever, then CH can answer that query without concerning CDS.

Let us consider that P_1 issues a query Q and it is in the vicinity of CH_1, hence CH_1 answers Q. If CH_1 does not have associated data-cache for Q, then it forwards Q to its group members in G1. Suppose, CH_4 hosts the data-cache regarding to Q, hence it replies to CH_1. Then CH_1 downloads a data-cache from CH_4 as CH_4 is far from P_1. Finally, CH_1 replies to P_1 from its new data-cache. Furthermore, if no any group member in G1 has data-cache associated with Q, then CH_1 forwards request for that data-cache to CM_1. Eventually, CM_1 does not have adequate data, then it forwards the request for data-cache about Q to CDS, buffers that data-cache and forwards back to CH_1. Once, CH_1 receives data-cache of Q, it stores in its cache memory for answering the future queries related to that data-cache in its vicinity. This is aligned to the real world scenario, where the query on current deceases are frequently sent by many people, hence the common data is utilized to perform quick query processing.

During the query processing, there may be a case when a QI may be in the vicinity of more than one CH of the same group or different groups. For example, P_{10} is in the vicinity of CH_1 and CH_2. Suppose, QI broadcasts a query Q, hence both of the cache-hosts CH_1 and CH_2 intercept Q and they try to respond it. Here, we assume that QI accepts the answer of Q from the cache-host, who responds first in query deadline time.

Query Processing in MobCache

In MobCache, query-issuer QI broadcasts a query Q, which is being intercepted by one of the nearby cache-hosts CHs. The encountered Q is being processed by one or more CHs through database processing related to Q on CH's data-cache (i.e., partial mobile database). The result of the Q has been sent back to QI and Q is marked as successful query in MobCache. If CH fails to answer Q, due to inadequate data-cache, then CH broadcasts Q to its nearby group members (i.e. other group-CHs). Upon receiving, other CHs process Q and CH with the relevant data-cache replies to original CH, who has forwarded Q.

Algorithm 1 MobCache: Cache-Host (CH)

```
begin

Input:  (a) $QI$ : Query-Issuer (b) $Q$ = Query (c) $CM$ : Cache-Manager

(1)  $CH$ receives $Q$ from $QI$
(2)  if ( $CH$ has relavant data)
(3)       $CH$ responds $QI$ with relavant data
(4)       if ($CH$ fails)
(5)            $CH$ forwards $Q$ in the group of $CH$
(6)       if ($CH$ has relavant data)
(7)            sends the requested data to the activated $CH$
(8)       else
(9)            retrieves data from $CM$
(10)           stores data in cache table
(10)$CH$ responds $QI$ with requested data

end
```

Algorithm 2 MobCache: Cache-Manager (CM)

```
begin

Input:  (a) $Q$ = Query (b) $CH$ : Cache-Host (c) $DB$ : Central Database Server

(1)  $CM$ receives data request from $CH$ for the $Q$
(2)  $CM$ sends request to $DB$
(3)  $CM$ retreives data from $DB$
(4)  sends data to $CH$

end
```

If CH successfully receives data-cache from one of its group CHs, then it sends Q's result back to QI and stores data-cache for similar future queries. If other CHs fail to send data-cache related to Q, then original CH forwards request to its relative CM (i.e., group leader). If CM has relevant data-cache, then it sends to CH, otherwise it obtains from CDS and forwards to CH.

Algorithms 1 and 2 represent the query execution procedures followed by cache-host and cache-manager respectively in MobCache. Notably, the query-issuer can have access to the only cache-hosts, while cache-hosts are directly associated with their relative cache-managers.

Groups in MobCache

A group is a collection of mobile peers (MPs). In MobCache, groups are formed on the basis of geographic vicinity. As discussed in architecture, each group has one group leader, which plays the role of cache-manager CM. Here, CM facilitates the storage of data-caches from Central Database Server (CDS) and distributes them to their group-members i.e., cache-hosts (CHs).

The cache-hosts are the mobile information service providers, who roam in the different region of the system towards providing quick information services to the mobile peers in their vicinity. In MobCache, the group-CHs are connected with each other via Bluetooth / WiFi / Low-cost Internet, while they are directly connected to CM via high network bandwidth through WiFi / high-speed Internet.

Each member in a group maintains data-cache in form of cache-table. Notably, it is application deterministic approach, thereby the cache-hosts and the cache-managers are decided based on application specifications. Observer that the cache-hosts and cache-managers along with their group descriptions are pre-specific and defined by the application system itself.

The data is to be cached on cache-host or cache-manager via push/pull mechanisms. Suppose, CH receives large number of queries on the data-cache, which is not available. In this case, CH pulls data-cache from CM, who may further pulls the required data-cache from CDS. Similarly, when CDS updates any data-cache, it pushes the updated data-cache to all CMs in MobCache. On updation, CMs forward this data-cache to their relevant CHs, who already host that data-cache. Finally, CHs refreshes their data-caches on periodic-basis towards serving up-to-date information to mobile peers in MobCache.

Lets consider an example of our medical information system across the market places. In market place, all the updated information about various medical product stores, medicines, treatments etc. are being kept on small central server at market place. This information covers locations, new arrivals, stocks and promotions of the medical products. Now, all CHs downloads this information from the central server in order to satisfy queries from the mobile users. Storing data on CH from central server results in low network traffic at server and mobile users are able to get answers quickly for their medical related queries, thereby increase in customer satisfaction. Moreover, a CH is able to communicate with other group-CHs in case if data is not available for the particular query. Here, central server at market place plays a role of CM for their particular application scenario.

In MobCache, all the groups are isolated i.e., at the same time one CH can not be a member of more than one groups. However, CH can change its group through central authority i.e., CM, who maintains the whole process of group formation and the member registration. Moreover, MobCache assumes that CH is only able to communicate with CM, other group CHs, and the mobile peers in its vicinity.

The motivation behind group formation is interest of mobile users. The groups can be formed using one of the following three techniques: (i) interest-based group, (ii) density-based group and (iii) event-based group. The interest-based groups are formed periodically according to diverse needs of mobile users as interest always varies over a period of time. While, density-based groups are formed on regular bases in high density areas. Moreover, event-based groups are formed eventually by considering data demands of mobile users available at particular event.

4. CACHING SCHEMES FOR MOBCACHE

The recent mobile devices are embedded with small storage capacity, which can be utilized to store partial data as per application specific requirements. This caching of frequently used data results in low network traffic at central database server (CDS), while reducing the communication cost. In MDS, the whole database can be distributed amongst the mobile peers in system and they are able to communicate with each other to collaboratively satisfy users' queries. Query processing using this local cache table results in faster query execution at low transaction cost.

This section discusses our proposed schemes for caching namely, **Cache-Less MDS (CLM)**, **Cache-Enabled MDS (CEM)** and **Group-based Caching in MDS (GCM)**. We assume the basic mechanism used for query processing in form of CLM scheme, which does not adapt any caching technique to store distributed databases. In contrast, subsequent schemes CEM and GCM incorporate the caching of data, which is supposed to be stored on local cache memory of mobile peers from CDS. CEM incorporates the single-level caching mechanism, while GCM has two-level caching mechanism with the notion of 'group-caching'. This increasing number of levels for caching results in reduction of network traffic at CDS and minimization the query processing time and communication cost as data is available through various levels of data-cache.

Cache-Less MDS (CLM)

In CLM, mobile peers directly communicate with CDS to obtain their query related data. Whenever, a mobile peer requires any data, it sends query Q to the CDS, which responds to mobile peer in form of data-table as answer of a query. CDS will also store the result data-table of Q in its own local memory. The subsequent queries, which are same as Q, from same or other mobile peers will be answered by CDS using the stored result of Q, thereby reducing computing time.

Notably, the communication cost does not get reduced as all mobile peers directly communicate with CDS, which results in high communication overhead in MDS. Moreover, increasing number of mobile peers' requests on CDS increases network traffic. The delaying on serving mobile peer on CDS results in time-out for a given query, thereby a query is failed as mobile peer is moving entity in network and it may not maintain constant communication with CDS.

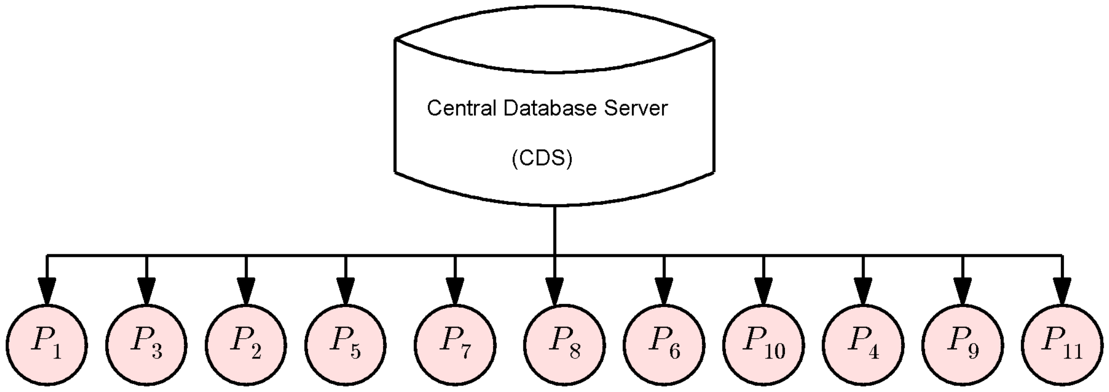

Figure 2: Illustrative example of CLM

Figure 2 illustrates an example of CLM, which consists mobile peers (P_1, ..., P_{11}) and CDS. Notably, all mobile peers have direct connections with CDS. Let us consider that P_1 issues a query Q_1 to CDS. In reply, CDS sends data-table T_1 to P_1 as answer of Q_1. Similarly, P_2 issues a query Q_2 directly to CDS; and so on. Suppose, CDS obtains the same data-table T_1 for Q_1 and Q_2, then it stores T_1 into its local-memory, thereby subsequent queries from other mobiles peers, who have similar queries like Q_1 and Q_2, can obtained answered quickly due to reduced computation time on CDS, but there is increase in communication cost and connection overhead on CDS. This results in delayed service by CDS, hence there is a high number of failures of subsequent queries.

Furthermore, few mobile peers may get connected with CDS continuously to download data for their sent query, due to low network bandwidth. In such cases, other mobile peers may go to wait state, due to delayed service by CDS. Such cases shows that, processing queries using CLM causes high communication cost and less query success rate as each mobile peer tries to access CDS independently.

Cache-Enabled MDS (CEM)

In contrast to CLM, CEM incorporates the notion of *caching* at mobile peers. In CEM, a mobile peer with high resources (e.g., memory, computation power, communication capability etc.) acts as cache-host (CH) and other mobile peers are querying users. Recall that CH is responsible for (i) serving data to the mobile peers in its vicinity, (ii) storing data-cache (i.e., partial database) into its local cache-memory in order to serve quickly and (iii) retrieving data from CDS based on received queries. In CEM, each mobile peer broadcasts query in mobile networks. This query is intercepted by its nearby CHs. If query related data is available on any one of the CHs, then that CH answers the query. In case, the intercepted CH does not have adequate data, it forwards query to CDS and obtain data. The received data is sent back to query issuing peer, while CH also stores that data for answering future query.

Notably, our application scenarios consider that the frequent similar queries from one or more peers are quickly answered by their nearby data-provider i.e., cache-hosts in MDS. For example, a medical information system, a medical representative (MR) acts as cache-host by hosting health related information, while they obtain the required data from central medical server. The mobile peers queries to these MRs towards obtaining health related information such as current virals with their symptoms and medicines, nearby medical-shops, medicine descriptions etc.

Interestingly, CEM provides more faster query response to its nearby mobile peers with respect to CLM, which does not have any caching-mechanisms to store the answer of frequently asked queries. CEM also reduced the overhead on CDS by computing queries at local, thereby reducing CDS's computing cost. Furthermore, CDS has fixed number of connections (i.e., connections with CHs only), thus to reduce communication links on CDS. Moreover, CDS and CHs have high network bandwidth connection (e.g., WiFi / 3G Internet), which provides faster cache-consistency and cache-synchronization. For the sake of concentrating on core concept, cache-consistency and cache-synchronization have been left as future work.

On other hand, CH provides faster query result to mobile peers in its vicinity from its local data-cache. The frequently asked queries have been cached on CH, thereby reducing communication cost with CDS. Also, low bandwidth mobile peers have been scattered across CHs in MDS, thereby each CH has few number of mobile peers to be served efficiently. In this way, CEM brings data-source to the mobile peers, for efficient and faster query processing.

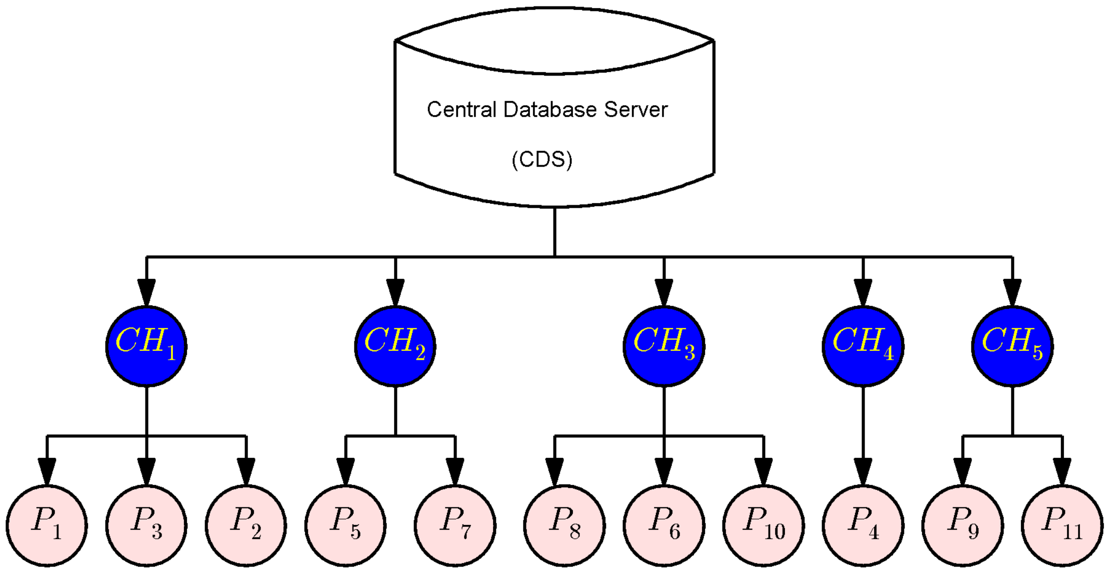

Figure 3: Illustrative example of CEM

As shown in Figure 3, we have cache-hosts (CH_1, ..., CH_5), who serve mobile peers (P_1, P_2, ..., P_{11}) according to their respective vicinity. For example, P_3 can be served by CH_1, as P_3 is in vicinity of CH_1. Suppose, P_3 issues a broadcast query Q, which is intercepted by CH_1 and CH_2, while consider that CH_1 is more closure to P_3 w.r.t. CH_2, thereby CH_1 can provide faster response to P_3. In this case, CH_1 sends result to P_3. If CH_1 does not have adequate data-cache, it sends request about Q to CDS and obtains required data. CH_1 stores that result as data-cache and responds Q. Notably, the vicinity of CHs change over a period of time, due to mobility of cache-hosts and mobile peers. This raises frequent network partitioning in MDS, but CEM ensures to provide consistent and efficient services by caching data at local cache-hosts, who are periodically synchronized with CDS.

Group-based Caching in MDS (GCM)

GCM advances the caching level to one step ahead w.r.t CEM. In GCM, the cache-hosts (CHs) have been grouped together to collaboratively provide better services towards query processing in MDS. As discussed in architecture of MobCache in Section 3, each CH is associated with a cache-manager (CM) in a group, hence CM acts as a group-leader, while CHs are group-members in a group of MobCache. In MobCache,CM is fully synchronized with regional-database (i.e., a partial data related to region in which CM belongs) from CDS, thereby it can almost act as a CDS for CH in MDS. For example, in medical information system, MRs act as CHs for GCM scheme, while regional-heads (RHs) are responsible as CMs. MR obtains the data-cache from RH as per the requirement from frequently asked queries, while RH synchronizes partial data related to its own region from CDS.

Suppose, a city like Delhi, which is supposed to be divided into several regions, which are managed by regional-heads, who have high capacity laptop or tablets to keep adequate information from CDS. Note that CMs (i.e., RHs in medical information system) are also mobile with less mobility but highly resourceful in terms of memory, computation and network bandwidth. Subsequently, CM distributes the information across their CHs in that area, while CHs roam across the streets, markets or such small-scale but high-density areas, thereby reducing response time to the queries raised by mobile peers. Furthermore, each CH collaboratively works in group i.e., if query Q received by CH_1 but it does not have data-cache related to Q, then CH_1 requests to other CHs into same group in which CH_1 belongs. If nobody in group has data-cache related to Q, then CH_1 forwards the request to its respective group leader CM, who obtains the latest information from CDS and pass on to CH_1. In this way, the whole-group collaboratively answers Q.

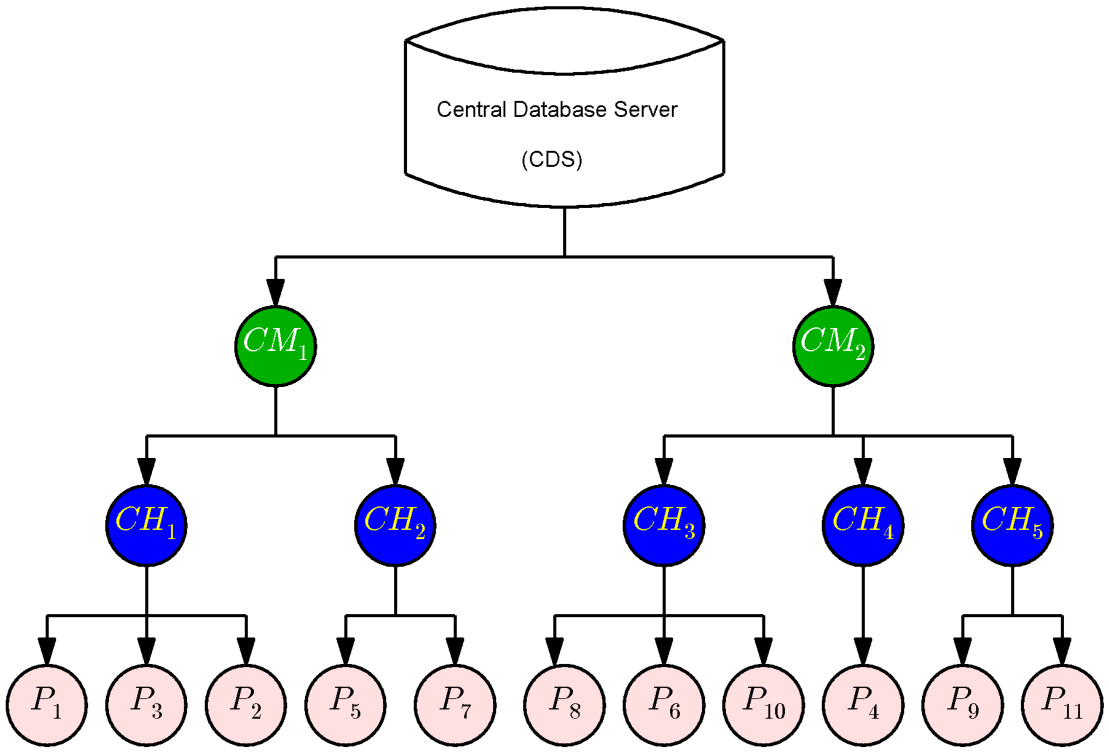

Figure 4: Illustrative example of GCM

Figure 4 represents the illustrative example of GCM scheme. Here, P_1 to P_{11} are the mobile peers, while CH_1 to CH_5 are the cache-hosts. $CH_1$and CH_2 belong to same group G1, which is controlled by cache-manager CM_1. Similarly, group G2 consists CM_2 as group-leader, while CH_3, CH_4 and CH_5 are the group-members.

5. PERFORMANCE STUDY

This section reports our performance evaluation by means of our own simulation based on architecture presented in Section 3. Our experiments consider a region with area of 1000 meters x 1000 meters. MPs in region move according to the Random Waypoint Model [Broch et al. 1998]. Communication range of cache-managers, cache-hosts and all MPs are a circle of 50, 20 and 5 meters radius respectively.

For each query Q, the query-issuer is selected randomly from among all MPs in the network. We set the query deadline time T_Q to 5 time units. Initial energy of an MP is selected to be randomly in the range of 90000 to 100000 energy units. In our simulation experiments, sending and receiving a message require 1.5 and 1 energy units respectively. Furthermore, we have consider the numbers of cache-managers, cache-hosts and query-issuers are Our all the experiments consider total 10,000 number of MPs, which includes cache-managers, cache-hosts and query-issuers.

Parameter	Default	Variations
Number of queries (N_Q) in 10^5	10	2, 4, 6, 8
Number of MPs (N_{MP}) in 10^3	10	2, 4, 6, 8
Cache-size (C_Z) in MB	6	2, 4, 8, 10
Number of queries per time unit (t.u.)	10	
Initial energy of an MP	90000 to 100000 energy units	
Speed of an MP	1 meter/s to 10 meters/s	

Table I: Parameters of performance study

Our performance metrics are **average response time (ART)**, **query success rate (QSR)** and **communication cost (MSG)**. We define query as **completed** if query issuer receives data within the query deadline time T_Q. Notably, the query issuer may fail to receive data due to reasons such as network partitioning and long waiting queue at higher level from where MP receives data.

We compute ART only for *completed* queries. $ART = \frac{1}{N_C} \sum_{q=1}^{N_C}(t_f - t_0)$, where t_0 is the query-issuing time, t_f is the time of the query result reaching the query-issuer, and N_C is the total number of *completed* queries. We compute ART in simulation time units (t.u.).

QSR is the ratio of total number N_C of completed queries to the total number N_Q of queries. $QSR = (\ N_C/N_Q\ ) \times 100$. We define MSG as the total number of messages incurred for query processing during the course of the experiment. Thus, $MSG = \sum_{q=1}^{N_Q} M_q$, where M_q is the number of messages incurred for the q^{th} query.

Performance of MobCache

We have conducted an experiment using the default values of the parameters in Table I. Figure 5 depicts the results. As N_Q increases, ART and MSG increase and QSR decreases for CLM and CEM due to high network connections at CDS. Furthermore, in CLM, frequent network disconnections are high due to mobility, thus QSR highly decreases as N_Q increases w.r.t. increased QSR for CEM and GCM. In GCM, query Q is intercepted and processed by its *nearby* available cache-host with data-cache related to Q, thereby reducing average response time as well as communication cost as compared to CEM, where missed cache is required to fetch each time from CDS by cache-hosts.

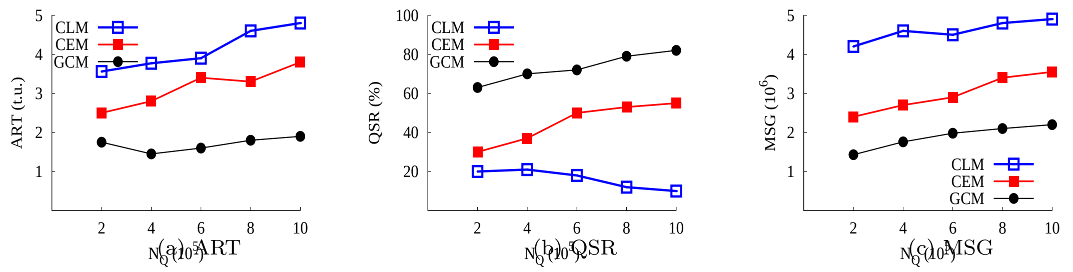

Figure 5: Performance of MobCache

Recall that, MSG is the cumulative measure of the messages over the course of the experiment. Hence, MSG increases over time for all of the three approaches as more queries are being processed. Due to query processing in vicinity of cache-hosts and cache-managers instead of CDS in GCM, communication cost is reduced as compared to CEM and CLM.

Effect of variations in N_{MP}

Figure 6 depicts the effect of variations in the number of mobile peers N_{MP}. As N_{MP} increases, ART and MSG increase for all three approaches due to larger network size. However, ART and

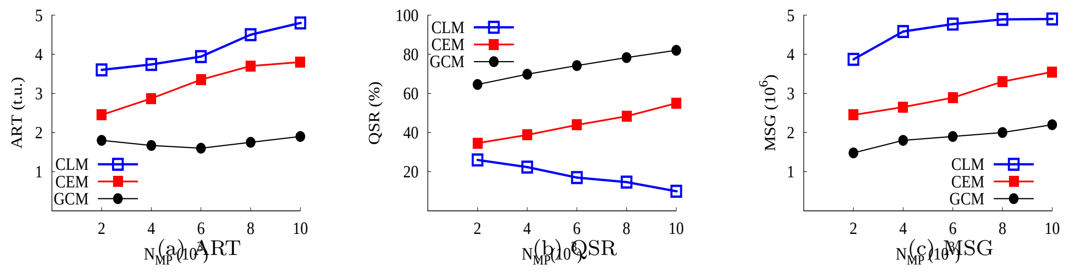

Figure 6: Effect of variations in N_{MP}

MSG increase highly for CLM and CEM w.r.t. GCM due to high network traffic at CDS for processing simultaneous large pool of queries. This further increases the query response time and communication cost. Moreover, a query is processed by cache-hosts available in vicinity in GCM, which results in reduced number of live connections at CDS, thereby QSR increases highly w.r.t. increased QSR for CEM. In CLM, each MP tries to access CDS directly, which causes high traffic on CDS, thereby increase in waiting time on CDS. Hence, due to more number of query failures causes decrease in overall QSR.

Effect of variations in C_Z

Caching highly depends on the cache-size in distributed networks. To observe similar effect of caching in mobile database system, we have performed the experiment with varying cache-sizes in MobCache.

Figure 7 represents the effect of variations in cache size C_Z. As CEM does not incorporate the notion of caching in MobCache, it exhibits the constant performance with varying in C_Z. As C_Z increases, CEM and GCM decrease due to more storage available at local mobile device,

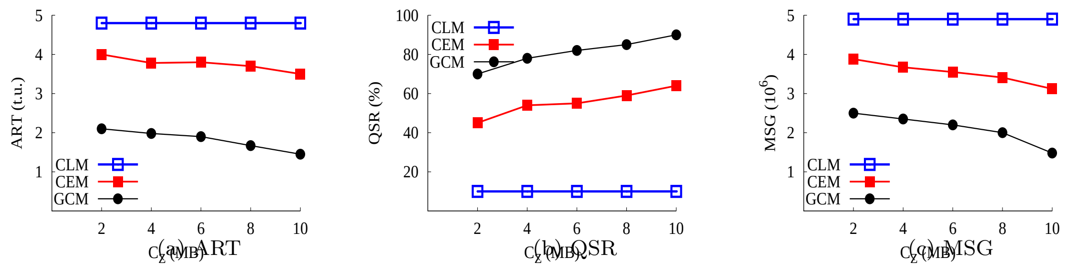

Figure 7: Effect of variations in C_Z

thereby more number of data-caches can be stored to quickly process and provide the answer to the upcoming and frequent queries. Furthermore, GCM exhibits better performance than CEM as group-level caching increases, thereby efficiently and quickly responds to nearby mobile peers.

As show in Figure 7a, QSR increases for CEM and GCM because increased possibility of storing more number of data-caches on cache-hosts results to answer queries within query-deadline time, thereby increase in successful queries in MobCache. Moreover, GCM has high performance than CEM due to increased capacity of overall data-caching of a group, which provides better accessibility and services to mobile peers in time, hence GCM performs better than CEM. While, as C_Z increases, MSG decreases drastically for GCM w.r.t. CEM as high data-caching causes reduced communication among cache-hosts and cache-managers, thereby decrease in MSG.

6. CONCLUSION

We have addressed group-based caching for Mobile Database System (MDS). In our proposed MobCache system, mobile peers store data in their local cache memory upon requirement by following group scenario. Storing of data in local cache memory results in minimization of communication cost, improvement of system performance by means of minimum connections at central database server and improvisation in query processing. We have proposed three mobile caching schemes namely, Cache-Less MDS (CLM), Cache-Enabled MDS (CEM) and Group-based Caching in MDS (GCM). Our performance evaluation shows that the MobCache is indeed effective in terms of reducing query response time and minimizing communication cost, thereby improving query processing in MDS. We have kept the cache-consistency and cache-synchronization policies in MobCache as future work.

REFERENCES

Amer, A., Long, D. D. E., and Burns, R. 2002. Group-based management of distributed file caches. In *Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on.* 525–534.

Barbará, D. 1999. Mobile Computing and Databases-A Survey. *IEEE TKDE 11,* 1, 108–117.

Broch, J., Maltz, D., Johnson, D., Hu, Y., and Jetcheva, J. 1998. A performance comparison of multi-hop wireless ad hoc network routing protocol. In *MOBICOM.*

Chan, D. and Roddick, J. F. 2003. Context-sensitive mobile database summarisation. In *ACSC.* 139–149.

Cheng, H., Gu, Z., and Ma, J. 2007. IntraCache: An Interest group-based P2P Web Caching System. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International.* 1–8.

Chow, C.-Y., Leong, H. V., and Chan, A. 2004. Peer-to-peer cooperative caching in a hybrid data delivery environment. In *Parallel Architectures, Algorithms and Networks, 2004. Proceedings. 7th International Symposium on.* 79 – 84.

Chow, C.-Y., Leong, H. V., and Chan, A. 2007. GroCoca: group-based peer-to-peer cooperative caching in mobile environment. *Selected Areas in Communications 25,* 1 (jan.), 179 –191.

Chow, C.-Y., Leong, H. V., and Chan, A. T. 2005. Distributed group-based cooperative caching in a mobile broadcast environment. In *Proceedings of the 6th international conference on Mobile data management.* ACM, 97–106.

Chung, Y. D. 2008. A cache invalidation scheme for continuous partial match queries in mobile computing environments. *Distributed and Parallel Databases 23,* 3, 207–234.

Dimokas, N., Katsaros, D., Tassiulas, L., and Manolopoulos, Y. 2011. High performance, low complexity cooperative caching for wireless sensor networks. *Wireless Networks 17,* 3, 717–737.

He, L., Sun, Y., and Zhang, C. 2010. Adaptive Group Based Replace Policy for high peroformance caching. In *Computer Science and Information Technology (ICCSIT)*. Vol. 9. 729–732.

Hu, H., Xu, J., Wong, W. S., Zheng, B., Lee, D. L., and Lee, W.-C. 2005. Proactive Caching for Spatial Queries in Mobile Environments. In *Data Engineering*. 403–414.

Janef, M. M., Parameswaran, R., Nadarajan, R., and Safar, M. 2008. PINE-guided cache replacement policy for location-dependent data in mobile environment. In *Pervasive Techniques Related to Assistive Environments*. 16:1–16:5.

Jung, I.-d., You, Y.-h., Lee, J.-h., and Kim, K. 2002. Broadcasting and caching policies for location-dependent queries in urban areas. In *WMC*. 54–60.

Kang, H. and Lim, S. 2001. Bandwidth-Conserving Cache Validation Schemes in a Moblie Database System. In *MDM*. Springer-Verlag, 121–132.

Kumar, A., Misra, M., and Sarje, A. K. 2007. A weighted cache replacement policy for location dependent data in mobile environments. In *System on Applied Computing*. 920–924.

Kumar, A., Sarje, A. K., and Misra, M. 2010. Prioritised Predicted Region based Cache Replacement Policy for location dependent data in mobile environment. *International Journal of Ad Hoc and Ubiquitous Computing 5,* 1, 56–67.

Kumar, V. 2006. *Mobile Database System*. Wiley - Interscience.

Li, D., Hu, M., and Chen, R. 2012. Quadtree-based management on semantic cache for mobile computing. *International Journal of Computer Applications in Technology 43,* 4, 393–399.

Li, Y. and Chen, I.-R. 2011. Adaptive per-user per-object cache consistency management for mobile data access in wireless mesh networks. *Journal of Parallel and Distributed Computing 71,* 7, 1034–1046.

Madria, S. K. and Bhowdrick, S. S. 2001. Mobile Data Management. *IEEE Potentials 20,* 4, 11–15.

Padhariya, N., Mondal, A., Goyal, V., Shankar, R., and Madria, S. K. 2011. EcoTop: An Economic Model for Dynamic Processing of Top-k Queries in Mobile-P2P Networks. In *Database Systems for Advanced Applications*. Springer, 251–265.

Padhariya, N., Mondal, A., Madria, S. K., and Kitsuregawa, M. 2013. Economic Incentive-based Brokerage Schemes for Improving Data Availability in mobile-P2P Networks. *Computer Communications 36,* 8, 861–874.

Peng, W.-C. and Chen, M.-S. 2005. Design and Performance Studies of an Adaptive Cache Retrieval Scheme in a Mobile Computing Environment. *IEEE TMC 4,* 1, 29–40.

Qureshi, K. A., Mohiuddin, S., Aziz-Uddin, A.-U., and Atique-Ur-Rehman, A.-U.-R. 2010. A strip-down database for modern information systems. In *Proc. of conf. on Computers*. World Scientific and Engineering Academy and Society (WSEAS), 81–88.

Sourlas, V., Paschos, G. S., Flegkas, P., and Tassiulas, L. 2009. Caching in content-based publish/subscribe systems. In *Global Telecommnication*. 1401–1406.

Sun, S., Zhou, X., and Shen, H. T. 2005. Semantic caching for multiresolution spatial query processing in mobile environments. In *Proc. of conf. on Advances in Spatial and Temporal DBs*. Springer-Verlag, 382–399.

Ting, I.-W. and Chang, Y.-K. 2013. Improved Group-based Cooperative Caching Scheme for Mobile Ad Hoc Networks. *Journal of Parallel and Distributed Computing 73,* 5, 595–607.

Tolia, N., Satyanarayanan, M., and Wolbach, A. 2007. Improving mobile database access over wide-area networks without degrading consistency. In *Mobile Systems, Applications and Services*. 71–84.

Zhang, J., Chu, Y., and Yang, J. 2006. A category on the cache invalidation for wireless mobile environments. In *APWeb*. 939–942.

Kshama Raichura is an assistant professor in computer science department of Shree M. & N. Virani Science college- Atmiya group of institutions, Gujarat, India. She is pursuing Ph. D. degree in computer science, under the guidance of Prof. Nilesh Padhariya, at Saurashtra University, Gujarat, India where she works on data management on mobile p2p system. She received her degree of BCA and M.Sc. (Info. Tech. & Com. App.) in the year 2009 and 2011 respectively from Saurashtra University, Gujarat, India.

Nilesh Padhariya is an associate professor of computer engineering at Atmiya Institute of Technology and Science, Gujarat, INDIA. He is perusing his Ph. D. degree in Computer Science at Indraprstha Institute of Information Technology, Delhi (IIIT-D), INDIA, where he works on mobile data management and economy-based incentive schemes for peer participation in mobile environment. He earned his M.Tech. degree in Computer Applications in 2006 from Indian Institute of Technology, Delhi (IIT-Delhi), one of the prestigious institutions of INDIA. His work addresses the efficient data management using effective economic incentive-based schemes in mobile ad hoc peer to peer (M-P2P) networks. It includes the dynamic query processing and the data replication in M-P2P networks using economic schemes. His work has been published at prestigious conferences and peer-reviewed journals from IEEE, ACM, ScienceDirect, etc. He has also received several grants for research and publications, which include SIGCOMM 2011, SRDS.